TALLER 6

Juan Camilo Moreno Bustos

Juan David Ortiz Gil

Facultad de ingeniería, Pontificia Universidad Javeriana

Análisis y Diseño de Software

Ing. Carlos Saldarriaga

Bogotá, Colombia

1. **Arquitectura utilizada:**

```
com.restaurant/
├── ui/                         # Capa de presentación
│   └── RestaurantConsoleApp
├── application/                # Capa de aplicación/servicios
│   ├── OrderService
│   ├── MenuService
│   └── ReportService
├── domain/                     # Capa de dominio
│   ├── model/                  # Entidades de dominio
│   │   ├── MenuItem
│   │   ├── Order
│   │   └── Customer
│   ├── service/                # Servicios de dominio
│   │   ├── PricingService
│   │   └── OrderProcessingService
│   └── repository/             # Interfaces de repositorio
│       ├── MenuRepository
│       └── OrderRepository
└── infrastructure/             # Capa de infraestructura
    ├── persistence/            # Implementaciones de repositorio
    │   ├── InMemoryMenuRepository
    │   └── InMemoryOrderRepository
    └── notification/           # Sistemas de notificación
        └── OrderNotifier
```

Codigo fuente:

```
restaurant-maven/src/main/java/com/restaurant/ui/RestaurantConsoleApp.java
package com.restaurant.ui;

import com.restaurant.infrastructure.persistence.*;
import com.restaurant.infrastructure.notification.OrderNotifier;
import com.restaurant.application.*;
import com.restaurant.domain.model.*;

import java.util.Scanner;

public class RestaurantConsoleApp {
    private final MenuService menuSvc;
    private final OrderService orderSvc;
    private final ReportService reportSvc;

    public RestaurantConsoleApp(){
        InMemoryMenuRepository menuRepo = new InMemoryMenuRepository();
        InMemoryOrderRepository orderRepo = new InMemoryOrderRepository();
        OrderNotifier notifier = new OrderNotifier();

        menuSvc = new MenuService(menuRepo);
        orderSvc = new OrderService(orderRepo, menuRepo, notifier);
        reportSvc = new ReportService(orderRepo);

        // seed menu
        menuSvc.addMenuItem(new MenuItem("M1","Hamburguesa",15000,"Main","Clásica"));
        menuSvc.addMenuItem(new MenuItem("B1","Gaseosa",5000,"Beverage","Refresco"));
    }

    private void showMenuOptions(){
        System.out.println("\n=== MENÚ ===");
        System.out.println("1. Ver menú");
        System.out.println("2. Crear plato");
        System.out.println("3. Modificar plato");
        System.out.println("4. Eliminar plato");
        System.out.println("0. Volver");
    }

    private void menuManagement(Scanner sc){
        String opt;
        do{
            showMenuOptions();
            System.out.print("Opción: ");
            opt = sc.nextLine();
            switch(opt){
                case "1" -> menuSvc.list().forEach(System.out::println);
                case "2" -> {
                    System.out.print("ID: "); String id = sc.nextLine();
                    System.out.print("Nombre: "); String name = sc.nextLine();
                    System.out.print("Precio: "); double price = Double.parseDouble(sc.nextLine());
                    System.out.print("Categoría: "); String cat = sc.nextLine();
                    System.out.print("Descripción: "); String desc = sc.nextLine();
                    menuSvc.addMenuItem(new MenuItem(id,name,price,cat,desc));
                }
                case "3" -> {
                    System.out.print("ID del plato a modificar: ");
                    String id = sc.nextLine();
                    MenuItem item = menuSvc.find(id);
                    if(item != null){
                        System.out.print("Nuevo nombre (" + item.getName() + "): ");
                        String name = sc.nextLine();
                        System.out.print("Nuevo precio (" + item.getPrice() + "): ");
                        double price = Double.parseDouble(sc.nextLine());
                        item.setName(name);
                        item.setPrice(price);
                        menuSvc.updateMenuItem(item);
                    } else {
                        System.out.println("Plato no encontrado");
                    }
                }
                case "4" -> {
                    System.out.print("ID del plato a eliminar: ");
                    menuSvc.removeMenuItem(sc.nextLine());
                }
            }
        }while(!opt.equals("0"));
    }

    public void run(){
        Scanner sc = new Scanner(System.in);
        String opt;
        do{
            System.out.println("\n=== Sistema Restaurante ===");
            System.out.println("1. Gestión de menú");
            System.out.println("2. Crear pedido");
            System.out.println("3. Añadir plato a pedido");
            System.out.println("4. Avanzar estado de pedido");
            System.out.println("5. Ver pedido");
            System.out.println("6. Total del pedido (con impuestos)");
            System.out.println("7. Ventas diarias");
            System.out.println("0. Salir");
            System.out.print("Opción: ");
            opt = sc.nextLine();

            try{
                switch(opt){
                    case "1" -> menuManagement(sc);
                    case "2" -> {
                        Customer cust = new Customer("C1","Cliente","Calle 1","3001234567");
                        System.out.println("ID del cliente: " + cust.getId());
                        System.out.println("Pedido creado con ID: " + orderSvc.createOrder(cust).getId());
                    }
                    case "3" -> {
                        System.out.print("ID Pedido: "); String oId = sc.nextLine();
                        System.out.print("ID Plato: "); String pId = sc.nextLine();
                        orderSvc.addItem(oId,pId);
                    }
                    case "4" -> {
                        System.out.print("ID Pedido: "); orderSvc.advanceStatus(sc.nextLine());
                    }
                    case "5" -> {
                        System.out.print("ID Pedido: ");
                        System.out.println(orderSvc.get(sc.nextLine()));
                    }
                    case "6" -> {
                        System.out.print("ID Pedido: ");
                        System.out.printf("Total (imp. incl.): $%.2f%n", orderSvc.total(sc.nextLine()));
                    }
                    case "7" -> {
                        System.out.printf("Ventas del día: $%.2f%n", reportSvc.dailySales());
                    }
                }
            }catch(Exception e){ System.out.println("Error: "+e.getMessage()); }

        }while(!opt.equals("0"));
        sc.close();
    }

    public static void main(String[] args){
        new RestaurantConsoleApp().run();
    }
}
restaurant-maven/src/main/java/com/restaurant/application/MenuService.java
package com.restaurant.application;

import com.restaurant.domain.repository.MenuRepository;
import com.restaurant.domain.model.MenuItem;
import java.util.List;

public class MenuService {
    private final MenuRepository repo;
    public MenuService(MenuRepository repo){ this.repo = repo; }

    public void addMenuItem(MenuItem item){ repo.save(item); }
    public List<MenuItem> list(){ return repo.findAll(); }
    public MenuItem find(String id){ return repo.findById(id); }

    public void updateMenuItem(MenuItem item){ repo.update(item); }
    public void removeMenuItem(String id){ repo.delete(id); }
}
restaurant-maven/src/main/java/com/restaurant/application/OrderService.java
package com.restaurant.application;

import com.restaurant.domain.repository.*;
import com.restaurant.domain.model.*;
import com.restaurant.domain.service.*;
import com.restaurant.infrastructure.notification.OrderNotifier;
import java.util.UUID;

public class OrderService {
    private final OrderRepository orderRepo;
    private final MenuRepository menuRepo;
    private final OrderProcessingService processingService;
    private final PricingService pricingService;

    public OrderService(OrderRepository orderRepo, MenuRepository menuRepo, OrderNotifier notifier){
        this.orderRepo = orderRepo;
        this.menuRepo = menuRepo;
        this.processingService = new OrderProcessingService(notifier);
        this.pricingService = new PricingService();
    }

    public Order createOrder(Customer customer){
        Order o = new Order(UUID.randomUUID().toString().substring(0,8), customer);
        orderRepo.save(o);
        return o;
    }

    public void addItem(String orderId, String menuItemId){
        Order o = orderRepo.findById(orderId);
        MenuItem item = menuRepo.findById(menuItemId);
        if(o != null && item != null){ o.addItem(item); }
    }

    public void applyDiscount(String orderId, Discount discount){
        Order o = orderRepo.findById(orderId);
        if(o != null){ o.setDiscount(discount); }
    }

    public void advanceStatus(String orderId){
        Order o = orderRepo.findById(orderId);
        if(o != null){ processingService.advance(o); }
    }

    public double total(String orderId){
        Order o = orderRepo.findById(orderId);
        return o == null ? 0 : pricingService.calculateTotal(o);
    }

    public Order get(String id){ return orderRepo.findById(id); }
}
restaurant-maven/src/main/java/com/restaurant/application/ReportService.java
package com.restaurant.application;

import com.restaurant.domain.repository.OrderRepository;
import com.restaurant.domain.model.*;
import com.restaurant.domain.service.PricingService;
import java.util.*;
import java.util.stream.Collectors;

public class ReportService {
    private final OrderRepository orderRepo;
    private final PricingService pricingService = new PricingService();
    public ReportService(OrderRepository orderRepo){ this.orderRepo = orderRepo; }

    public double dailySales(){
        return orderRepo.findAll().stream()
```

```java
                .mapToDouble(o -> pricingService.calculateTotal(o))
                .sum();
    }

    public Map<String, Long> popularItems(){
        return orderRepo.findAll().stream()
                .flatMap(o -> o.getItems().stream())
                .collect(Collectors.groupingBy(MenuItem::getName, Collectors.counting()));
    }
}
restaurant-maven/src/main/java/com/restaurant/domain/model/MenuItem.java
package com.restaurant.domain.model;

public class MenuItem {
    private String id;
    private String name;
    private double price;
    private String category;
    private String description;

    public MenuItem(String id, String name, double price, String category, String description) {
        this.id = id;
        this.name = name;
        this.price = price;
        this.category = category;
        this.description = description;
    }

    public String getId() { return id; }
    public String getName() { return name; }
    public double getPrice() { return price; }
    public String getCategory() { return category; }
    public String getDescription() { return description; }

    public void setName(String name){ this.name = name; }
    public void setPrice(double price){ this.price = price; }
    public void setCategory(String category){ this.category = category; }
    public void setDescription(String description){ this.description = description; }

    @Override     public String toString() {
        return String.format("%s [%s] - $%.2f", name, category, price);
    }
}
restaurant-maven/src/main/java/com/restaurant/domain/model/Discount.java
```

```
package com.restaurant.domain.model;

@FunctionalInterface public interface Discount {
    double calculate(double amount);
}
restaurant-maven/src/main/java/com/restaurant/domain/model/FixedDiscount.java
package com.restaurant.domain.model;

public class FixedDiscount implements Discount {
    private final double discount;
    public FixedDiscount(double discount){ this.discount = discount; }
    @Override public double calculate(double amount){
        return Math.max(0, amount - discount);
    }
    @Override public String toString(){
        return String.format("Descuento fijo $%.2f", discount);
    }
}
restaurant-maven/src/main/java/com/restaurant/domain/model/PercentDiscount.java
package com.restaurant.domain.model;

public class PercentDiscount implements Discount {
    private final double percent;
    public PercentDiscount(double percent){ this.percent = percent; }
    @Override public double calculate(double amount){
        return amount * (1 - percent/100.0);
    }
    @Override public String toString(){
        return String.format("Descuento %.0f%%", percent);
    }
}
restaurant-maven/src/main/java/com/restaurant/domain/model/Customer.java
package com.restaurant.domain.model;

public class Customer {
    private final String id;
    private final String name;
    private final String address;
    private final String phone;

    public Customer(String id, String name, String address, String phone){
        this.id = id;
        this.name = name;
        this.address = address;
```

```java
        this.phone = phone;
    }

    public String getId(){ return id; }
    public String getName(){ return name; }
    public String getAddress(){ return address; }
    public String getPhone(){ return phone; }

    @Override public String toString(){
        return String.format("%s (%s)", name, phone);
    }
}
```
restaurant-maven/src/main/java/com/restaurant/domain/model/Order.java
```java
package com.restaurant.domain.model;

import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;

public class Order {
    public enum Status { RECIBIDO, PREPARACION, LISTO, ENTREGADO }

    private final String id;
    private final Customer customer;
    private final List<MenuItem> items = new ArrayList<>();
    private Status status = Status.RECIBIDO;
    private final LocalDateTime date = LocalDateTime.now();
    private Discount discount;

    public Order(String id, Customer customer){
        this.id = id;
        this.customer = customer;
    }

    // getters
    public String getId(){ return id; }
    public Customer getCustomer(){ return customer; }
    public List<MenuItem> getItems(){ return items; }
    public Status getStatus(){ return status; }
    public LocalDateTime getDate(){ return date; }

    public void addItem(MenuItem item){ items.add(item); }
    public void setDiscount(Discount discount){ this.discount = discount; }
    public double subtotal(){
        return items.stream().mapToDouble(MenuItem::getPrice).sum();
    }

    public double totalAfterDiscount(){
        double amount = subtotal();
        return discount != null ? discount.calculate(amount) : amount;
    }

    public void nextStatus(){
        switch(status){
            case RECIBIDO -> status = Status.PREPARACION;
            case PREPARACION -> status = Status.LISTO;
            case LISTO -> status = Status.ENTREGADO;
            default -> {}
        }
    }

    @Override public String toString(){
        return String.format("Pedido #%s [%s] - Subtotal $%.2f", id, status, subtotal());
    }
}
```

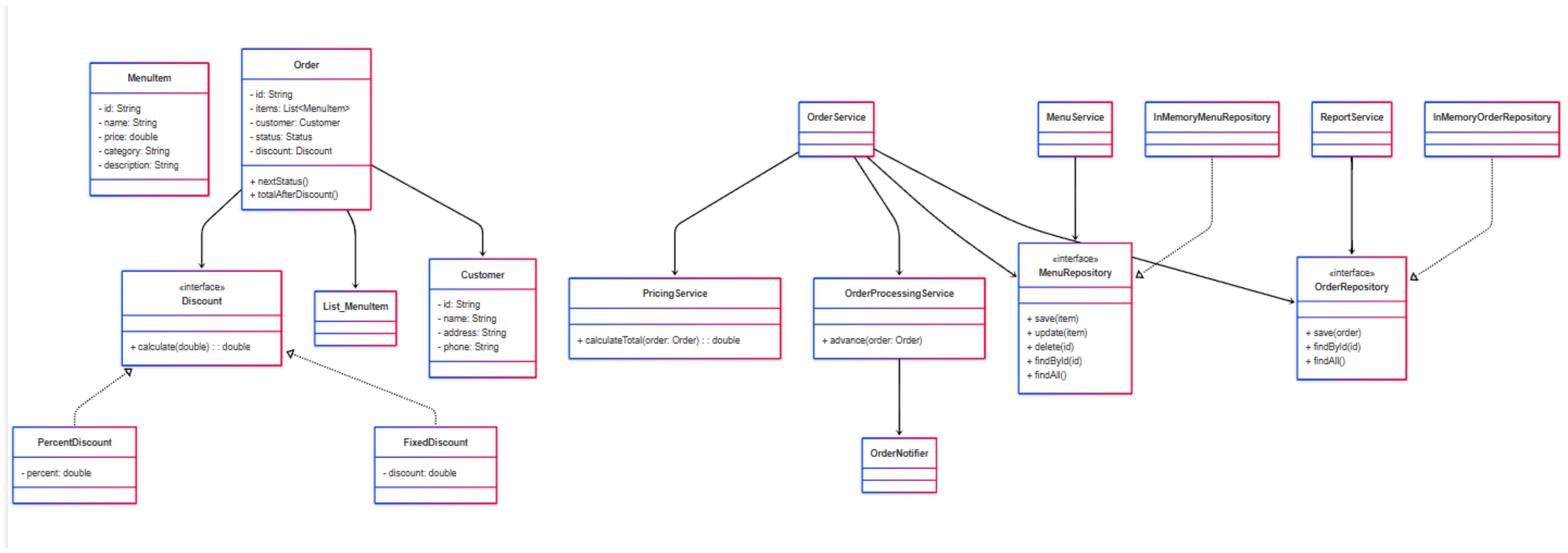
restaurant-maven/src/main/java/com/restaurant/domain/service/PricingService.java

```java
package com.restaurant.domain.service;

import com.restaurant.domain.model.Order;

public class PricingService {
    public static final double TAX_RATE = 0.19; // 19 %

    public double calculateTotal(Order order){
        double afterDiscount = order.totalAfterDiscount();
        return afterDiscount * (1 + TAX_RATE);
    }
}
```

restaurant-maven/src/main/java/com/restaurant/domain/service/OrderProcessingService.java

```java
package com.restaurant.domain.service;

import com.restaurant.domain.model.Order;
import com.restaurant.infrastructure.notification.OrderNotifier;

public class OrderProcessingService {
    private final OrderNotifier notifier;
    public OrderProcessingService(OrderNotifier notifier){ this.notifier = notifier; }
    public void advance(Order order){
        order.nextStatus();
        notifier.notifyStatus(order);
    }
}
restaurant-maven/src/main/java/com/restaurant/domain/repository/MenuRepository.java
package com.restaurant.domain.repository;

import java.util.List;
import com.restaurant.domain.model.MenuItem;

public interface MenuRepository {
    void save(MenuItem item);
    MenuItem findById(String id);
    List<MenuItem> findAll();
    void update(MenuItem item);
    void delete(String id);
}
restaurant-maven/src/main/java/com/restaurant/domain/repository/OrderRepository.java
package com.restaurant.domain.repository;

import java.util.List;
import com.restaurant.domain.model.Order;

public interface OrderRepository {
    void save(Order order);
    Order findById(String id);
    List<Order> findAll();
}
restaurant-maven/src/main/java/com/restaurant/infrastructure/persistence/InMemoryMenuRepository.java
package com.restaurant.infrastructure.persistence;

import com.restaurant.domain.repository.MenuRepository;
import com.restaurant.domain.model.MenuItem;
import java.util.*;

public class InMemoryMenuRepository implements MenuRepository {
    private final Map<String, MenuItem> db = new HashMap<>();

    @Override public void save(MenuItem item){ db.put(item.getId(), item); }
    @Override public MenuItem findById(String id){ return db.get(id); }
    @Override public List<MenuItem> findAll(){ return new ArrayList<>(db.values()); }
    @Override public void update(MenuItem item){ db.put(item.getId(), item); }
    @Override public void delete(String id){ db.remove(id); }
}
restaurant-maven/src/main/java/com/restaurant/infrastructure/persistence/InMemoryOrderRepository.java
package com.restaurant.infrastructure.persistence;

import com.restaurant.domain.repository.OrderRepository;
import com.restaurant.domain.model.Order;
import java.util.*;

public class InMemoryOrderRepository implements OrderRepository {
    private final Map<String, Order> db = new HashMap<>();
    @Override public void save(Order o){ db.put(o.getId(), o); }
    @Override public Order findById(String id){ return db.get(id); }
    @Override public List<Order> findAll(){ return new ArrayList<>(db.values()); }
}
restaurant-maven/src/main/java/com/restaurant/infrastructure/notification/OrderNotifier.java
package com.restaurant.infrastructure.notification;

import com.restaurant.domain.model.Order;

public class OrderNotifier {
    public void notifyStatus(Order order){
        System.out.println("Notificación: Pedido " + order.getId() + " ahora está " + order.getStatus());
    }
}
```

2. Diagrama UML



3. Descripción de los Patrones de Diseño Utilizados

- Strategy – Discount
    - Permite cambiar dinámicamente la forma en que se calcula el descuento aplicado a un pedido.
        - Interfaz: Discount
        - Implementaciones: FixedDiscount, PercentDiscount
        - Uso: Se asigna a través de Order.setDiscount(...) y se aplica en el cálculo total.

- State (implícito) – Order.Status
    - Modela el ciclo de vida de un pedido desde que es recibido hasta que se entrega.
        - Estados: RECIBIDO → PREPARACION → LISTO → ENTREGADO
        - La transición está implementada dentro del método Order.nextStatus().

- Repository – MenuRepository, OrderRepository
    - Define interfaces para desacoplar la lógica de dominio del mecanismo de persistencia.
        - Las interfaces se encuentran en el paquete domain.repository
        - Las implementaciones usan almacenamiento en memoria (InMemoryMenuRepository, InMemoryOrderRepository)

- Notificación desacoplada – OrderNotifier
    - Permite notificar el cambio de estado de un pedido sin acoplarse a la capa de presentación.
        - Se inyecta como dependencia en OrderProcessingService
        - Imprime una notificación simple cuando un pedido cambia de estado.

4. Conjunto de Pruebas Unitarias Básicas

Ubicación:
test/com/restaurant/

- OrderStateTest - Verifica las transiciones de estado de un pedido:
  - |De RECIBIDO a PREPARACION, luego a LISTO y finalmente ENTREGADO.

- PricingServiceTesT - Comprueba que el total del pedido:
  - Aplica correctamente el descuento (porcentaje o fijo).
  - Añade automáticamente el IVA del 19 % sobre el monto resultante.

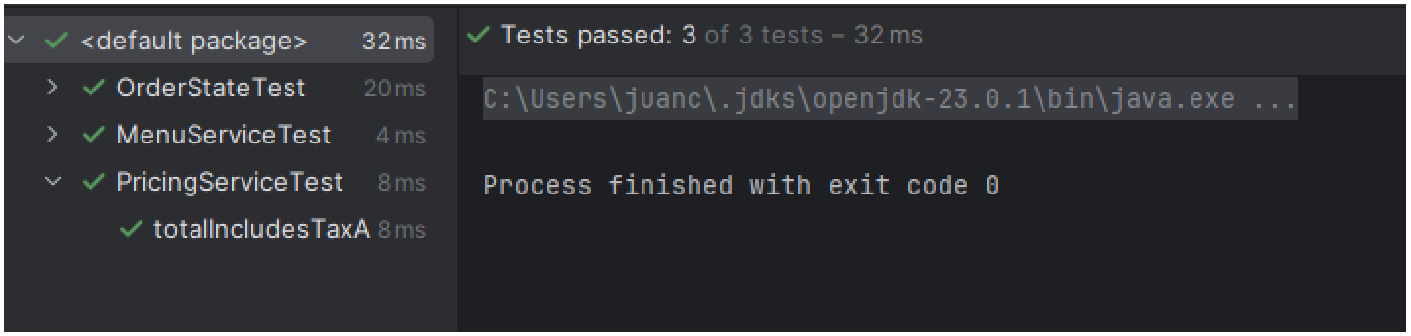
- MenuServiceTest - Evalúa las operaciones básicas sobre los platos del menú:
  - Crear un nuevo plato.
  - Modificar el precio de un plato existente.
  - Eliminar un plato del sistema.

```
 package com.restaurant;

import com.restaurant.application.MenuService;
import com.restaurant.domain.model.MenuItem;
import com.restaurant.infrastructure.persistence.InMemoryMenuRepository;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

public class MenuServiceTest {

   @Test
```

```
    void crudOperations() {
        var repo = new InMemoryMenuRepository();
        var service = new MenuService(repo);

        MenuItem item = new MenuItem("P1","Pizza",12000,"Main","4 quesos");
        service.addMenuItem(item);
        assertEquals(1, service.list().size());

        // modify
        item.setPrice(13000);
        service.updateMenuItem(item);
        assertEquals(13000, service.find("P1").getPrice());

        // delete
        service.removeMenuItem("P1");
        assertTrue(service.list().isEmpty());
    }
}
```

```
package com.restaurant;

import com.restaurant.application.MenuService;
import com.restaurant.domain.model.MenuItem;
import com.restaurant.infrastructure.persistence.InMemoryMenuRepository;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

public class MenuServiceTest {

    @Test
    void crudOperations() {
        var repo = new InMemoryMenuRepository();
        var service = new MenuService(repo);

        MenuItem item = new MenuItem("P1","Pizza",12000,"Main","4 quesos");
        service.addMenuItem(item);
        assertEquals(1, service.list().size());

        // modify
        item.setPrice(13000);
        service.updateMenuItem(item);
        assertEquals(13000, service.find("P1").getPrice());

        // delete
        service.removeMenuItem("P1");
        assertTrue(service.list().isEmpty());
    }
}



package com.restaurant;

import com.restaurant.domain.model.*;
import com.restaurant.domain.service.PricingService;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

public class PricingServiceTest {

    @Test
    void totalIncludesTaxAndDiscount() {
        Order order = new Order("1", new Customer("C","X","Dir","300"));
        order.addItem(new MenuItem("M","Burger",10000,"Main",""));
        order.setDiscount(new PercentDiscount(10)); // 10 %

        double total = new PricingService().calculateTotal(order);
        assertEquals(10710, total, 0.1);
    }
}
```

Evidencia Test:



```
<default package>        32 ms
  OrderStateTest         20 ms
  MenuServiceTest         4 ms
  PricingServiceTest      8 ms
    totalIncludesTaxA     8 ms
```

```
Tests passed: 3 of 3 tests – 32 ms

C:\Users\juanc\.jdks\openjdk-23.0.1\bin\java.exe ...

Process finished with exit code 0
```